Predictive Analysis for E-Commerce website and product Recommendation System

Devyani Bothra, Jinansi Thakkar and Gauri Yadav
Northeastern University
Department of Information Systems

Abstract. Recommendation System is the knowledge discovery techniques and use of statistical to deliver users with personalized content and service. It is used to solve the interaction with the customers which are targeted to provide product recommendation Issue. We found this a challenging task to build a recommendation system We found that more than half of the recommendation approaches applied content-based filtering (55 %). Collaborative filtering was applied by only 18 % of the reviewed approaches, and graph-based recommendations by 16 %. Other recommendation concepts included stereotyping, item-centric recommendations, and hybrid recommendations. In this project, we attempt to understand different kind of recommendation systems and compare their performance on E-commerce dataset. First, it remains unclear which recommendation concepts and approaches are most promising. Sometimes content-based filtering performs well or sometimes collaborative filtering. We tried to research deep into filtering techniques and apply for our dataset to evaluate recommender system adequately. We have used Collaborative Algorithms like Alternating Least Square(ALS), Random Forest Classifier, Naïve Bayes, K-means Clustering, Decision Tree Regressor and Content-based Algorithms like Tf-Idf method and Bag of Words model. We have also implemented Recurrent Neural Network for our recommender system.

I. INTRODUCTION

Traditionally, people used to buy product from the stores. But now -a- days people prefect online Shopping. With the massive adoption of internet/web as an e-commerce platform, led to change in the way business interact with their users. Recommendation System is the method to provide customers with suggestions about the product they could buy.

In this paper we have taken a dataset from data.world and we tried to provide recommendation to our users for several products having similar taste as of our other customer. Our dataset had relevant columns like UserID, ProductID, ProductDescription, Rating, DoReccomended Columns to do predictive analysis. We applied various machine learning and deep learning algorithm based on regression and clustering and found Collaborative filtering and ALS Method to be most suitable algorithms for Recommendation Systems

The recommendation system produced basically by two types 1. Collaborative Filtering (Item based, and User based) 2. Content- based Filtering. In collaborative filtering we find how many of users or items in the data are similar to other user and using correlation and cosine similarity find item for others. While the Content- based recommendation works on the rating/data provided explicitly or implicitly by the user. BBR provides recommendation which are highly personalized by matching user interest and description. We use standard Machine learning algorithms like SVM, decision Tress and Logistic Regression for making prediction in CBR

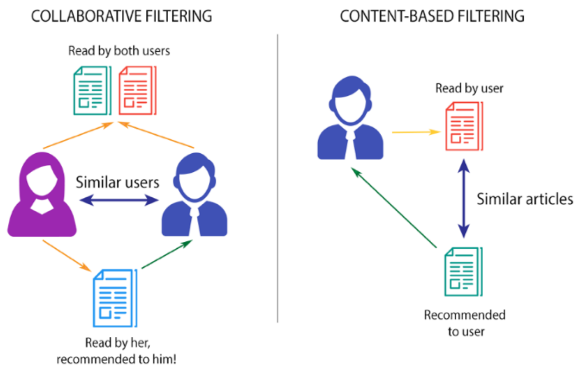


Fig. 1. Process of Recommendation System

Recommendation System is the knowledge discovery techniques and use of statistical to deliver users with personalized content and service. It is used to solve the interaction with the customers which are targeted to provide product recommendation Issue.

Machine learning algorithm are commonly used to due meaningful analysis on and allow computer to learn human behavior or nature and improve the performance of new task on old analysis. Additional benefit of recommendation system is:

1. **Convert Visitor to Buyer:** Recommendation systems help consumers find items that best fit the customers interests and inclinations and many times these lead to unplanned purchases driven by the buyer just because of the suggestion made.

2. **Increase in Cross-sell:** Giving Recommendation for products helps improve in cross selling by suggesting more products or services to customers. If the suggestions are perfect, the average order size increases.

3. **Create Value- added relationship:** In a competitive world where everything is one click away, building customer-loyalty becomes an essential aspect of business. Each time a customer visits a website, the system “learns” more about that customer’s preferences and interests and is increasingly able to operationalize this information to e.g. personalize what is offered. By providing each customer with an increasingly relevant experience, a corresponding improvement in the likelihood of that customer returning is achieved.

II. EVALUATE ALGORITHM ACCURACY

Algorithm error

The error in algorithm helps you to find the most suitable algorithm for your dataset and helps to obtain intended results. In this system we used few statistical measures to find out the best algorithm and highest correlation. Some of these are listed below.

1. R-squared: It is a statistical measure of how close the data are to the fitted regression line [4]. The higher the value, the better the model fits the dataset.

2.Root Mean Square Error(RMSE): It is a frequently used measure of the differences between values (sample and population values) predicted by a model or an estimator and the values actually observed. The lower the value, the better the model fits the dataset.

3.Loss Function: It is used for parameter estimation that maps an event or values of one or more variables onto a real variable intuitively representing some "cost" associated with the event. The main problem is to minimize a loss function.

Dataset

The dataset which we are using is from data.world with 100k rows and 26 columns. We have an E-commerce dataset with details of Product like EAN number, product Id, name, category, date Added, date Updated, rating, product URL, product manufacturer and rating. Dataset also contains user like userid, username, user review for product, user rating for product, user city, user purchase and user recommendation for product. User recommendation columns consist of True/False value to show if the user recommends a product or not. Rating consist of value between 1 to 5. The dataset containing dataset of E-commerce data having approximately 72K rows and 26 columns after filtering and removing rows which are cannot be used by performing Exploratory Data Analysis and Data Cleaning. In Algorithms used, we split dataset into two partitions: Test and train dataset by sampling in the ratios 20% and 80% respectively.

- Provide related items for the users from relevant and irrelevant collection of items.
- Given a data of items purchased by Customers, we are trying to predict items for the user which can be useful for them based on user's past purchase history and location of the user.
- Demand forecasting can also be made of items according to the item sold in a country/continent.

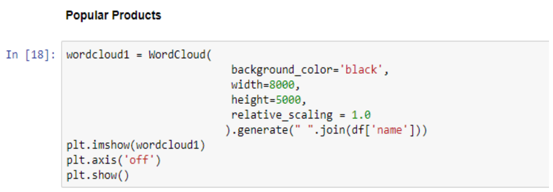
Data Processing

We start by filtering data and defining which columns will be used for recommendation algorithms. We have not used the rows which have any kind of reviews or purchases and no recommendations.

Feature Selection

In this system we used selected the most relative feature by generating the score of each feature and selecting the most relevant one. We have columns like doRecommed(True/False), didPurchase(True/False), rating, reviews which are very much effective in our recommender systems.

The below figure shows the words which are most repeated in comments. Using this we can build content-based recommendation system.



Popular Products

```
In [18]: wordcloud1 = WordCloud(
                                  background_color='black',
                                  width=8000,
                                  height=5000,
                                  relative_scaling = 1.0
                                 ).generate(" ".join(df['name']))
         plt.imshow(wordcloud1)
         plt.axis('off')
         plt.show()
```

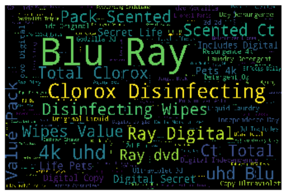


Fig. 2. Word Cloud

III. MACHINE LEARNING ALGORITHMS

1. Collaborative Filtering:

a.) Alternating Least Square (ALS) Algorithms:

We have used userId, rating and customerId for recommendation of product in ALS Algorithm using pyspark library in Apache Spark. ALS algorithm considers similarity between user's taste for product as well as user's past purchases.

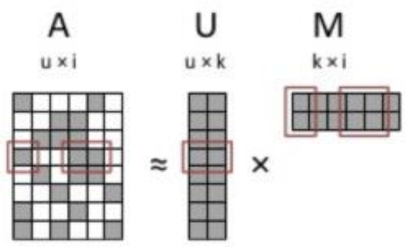
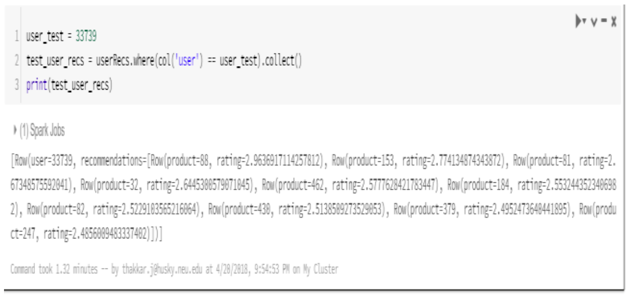


Fig. 3. ALS Algorithm Working



```
user_test = 33739
test_user_recs = userRecs.where(col('user') == user_test).collect()
print(test_user_recs)
```

```
▸ (1) Spark Jobs
[Row(user=33739, recommendations=[Row(product=88, rating=2.9636917114257812), Row(product=153, rating=2.774134874343872), Row(product=81, rating=2.67348575592041), Row(product=32, rating=2.6445388579071045), Row(product=462, rating=2.5777628421783447), Row(product=184, rating=2.5532443523406982), Row(product=82, rating=2.5229183565216064), Row(product=430, rating=2.5138589273529053), Row(product=379, rating=2.4952473640441895), Row(product=247, rating=2.4856089483337402)])]
Command took 1.32 minutes -- by thakkar.j@husky.neu.edu at 4/28/2018, 9:54:53 PM on My Cluster
```

Fig. 4. ALS Algorithm results

The above figure shows the recommended products for user having Id = 33739.

ALS rotates between fixing U and M. When U is fixed system computes M by solving least square problems per item and vice versa.

Source Code :
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/1619801862281876/1879698768322425/1006859747200966/latest.html

b.) K-Means Algorithm:

The K-means is a simple clustering algorithm using pyspark library in Apache Spark. 12 Clusters are formed with their centroids shown. Clustering is formed according to the similar user's choices and their location.

Source code :
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/1619801862281876/3577103074595437/1006859747200966/latest.html

2. Classification Algorithm:

a.) Naïve Bayes Algorithm:

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set.

$$P(c|x) = \frac{P(x|c)\, P(c)}{P(X)}$$

Where,

P(*c*|*x*) = posterior probability of *class* (c, *target*) given *predictor* (x, *attributes*).

P(*c*) = prior probability of *class*.

P(*x*|*c*) = likelihood which is the probability of *predictor of* given *class*.

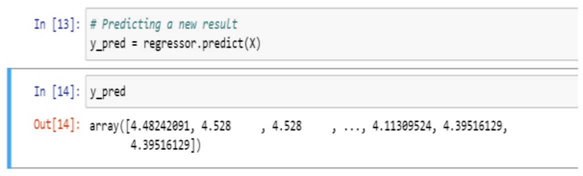
P(*x*) = prior probability of *predictor*.

Source code:
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/1619801862281876/3577103074595437/1006859747200966/latest.html

b.) Decision Tree Regressor:

The decision tree is used to fit a sine curve with addition noisy observation. As a result, it learns local linear regressions approximating the sine curve.

Using max_depth = 5, we can get more accuracy as if we take less depth for regression tree, the accuracy decreases.



```
In [13]: # Predicting a new result
         y_pred = regressor.predict(X)

In [14]: y_pred

Out[14]: array([4.48242091, 4.528     , 4.528     , ..., 4.11309524, 4.39516129,
                4.39516129])
```

Fig. 5. Decision Tree Results

Figure 5 shows the predicted values of y_test dataset. After training the dataset with training dataset, we test the algorithm for y_test dataset.

Source Code:
https://github.com/Jinansi/PythonProjects/blob/master/ADS_Project/DecisionTree_RandomForest.ipynb

c.) ***Random Forest Classifier:***
It is method for classification and regression and other tasks, that operate by constructing a multitude of decision tree at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

Features for this algorithm are didPurchase and Rating column. We have calculated the importance score of each feature which affects the recommendation value of product.

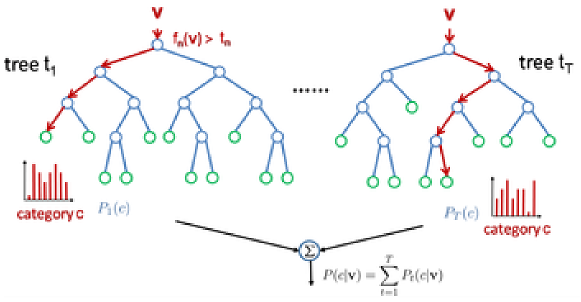
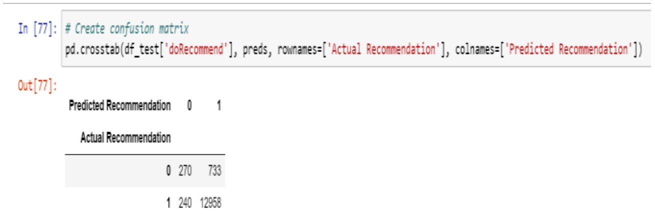


Fig. 6. Random Forest Classifier

The below graph shows the crosstab of actual and predicted value of products which shows if user recommends the product or not. 0 represents that the product is not recommended and 1 represents that product is recommended.



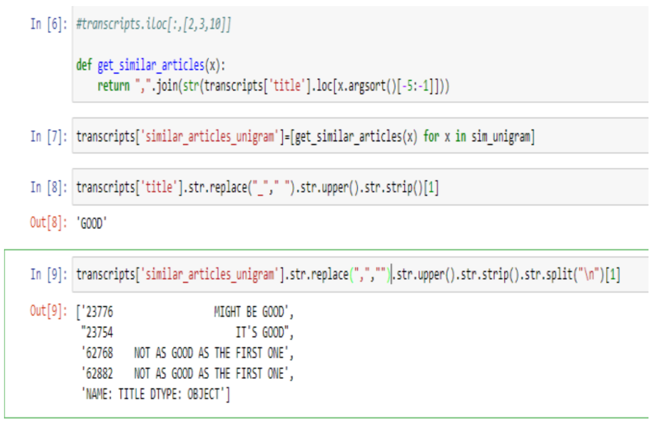
```
In [77]: # Create confusion matrix
         pd.crosstab(df_test['doRecommend'], preds, rownames=['Actual Recommendation'], colnames=['Predicted Recommendation'])

Out[77]:
```

Predicted Recommendation	0	1
Actual Recommendation		
0	270	733
1	240	12958

Fig. 7. Cross tab using Random Forest Classifier

Source Code:
https://github.com/Jinansi/PythonProjects/blob/master/ADS_Project/DecisionTree_RandomForest.ipynb

3. Content-based Filtering using TF-IDF:
Tf–idf is short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.



```
In [6]: #transcripts.iloc[:,[2,3,10]]

        def get_similar_articles(x):
            return ",".join(str(transcripts['title'].loc[x.argsort()[-5:-1]]))

In [7]: transcripts['similar_articles_unigram']=[get_similar_articles(x) for x in sim_unigram]

In [8]: transcripts['title'].str.replace("_"," ").str.upper().str.strip()[1]

Out[8]: 'GOOD'

In [9]: transcripts['similar_articles_unigram'].str.replace(",","").str.upper().str.strip().str.split("\n")[1]

Out[9]: ['23776                MIGHT BE GOOD',
         "23754                    IT'S GOOD",
         '62768    NOT AS GOOD AS THE FIRST ONE',
         '62882    NOT AS GOOD AS THE FIRST ONE',
         'NAME: TITLE DTYPE: OBJECT']
```

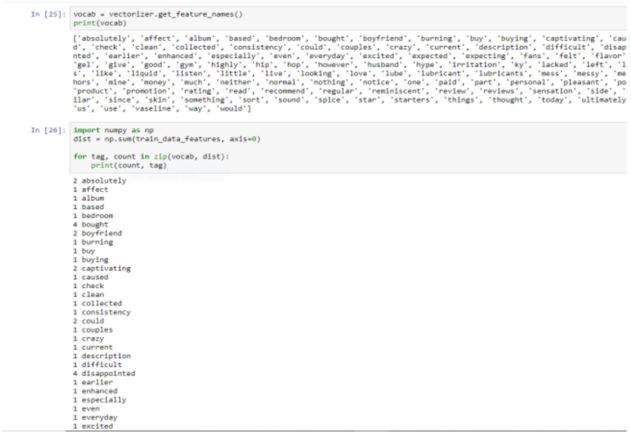
Fig. 8. Content-based filtering using TF-IDF

Source Code:
https://github.com/Jinansi/PythonProjects/blob/master/ADS_Project/Content%20Based%20using%20TF-IDF.ipynb

4. Natural Language Processing (NLP):

a.) Bag of Words:

In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. In the given dataset we considered the column "text" which contains the reviews given by a user for a product to achieve this concept. Using regular expression, we get a search pattern and clean the data by removing the punctuation marks and special characters. By importing stop words from natural language toolkit, we later try to segregate the important and frequent used terms from the common words of English dictionary. We used Count Vectorizer to count the number of times a word occurs in a corpus. Hence, we display a list of words with their counts which are important and frequent in a corpus.



```
In [25]: vocab = vectorizer.get_feature_names()
         print(vocab)

         ['absolutely', 'affect', 'album', 'based', 'bedroom', 'bought', 'boyfriend', 'burning', 'buy', 'buying', 'captivating', 'cau
         d', 'check', 'clean', 'collected', 'consistency', 'could', 'couples', 'crazy', 'current', 'description', 'difficult', 'disap
         nted', 'earlier', 'enhanced', 'especially', 'even', 'everyday', 'excited', 'expected', 'expecting', 'fans', 'felt', 'flavor'
         'gel', 'give', 'good', 'gym', 'highly', 'hip', 'hop', 'however', 'husband', 'hype', 'irritation', 'ky', 'lacked', 'left', 'l
         s', 'like', 'liquid', 'listen', 'little', 'live', 'looking', 'love', 'lube', 'lubricant', 'lubricants', 'mess', 'messy', 'me
         hors', 'mine', 'money', 'much', 'neither', 'normal', 'nothing', 'notice', 'one', 'paid', 'part', 'personal', 'pleasant', 'po
         'product', 'promotion', 'rating', 'read', 'recommend', 'regular', 'reminiscent', 'review', 'reviews', 'sensation', 'side', '
         ilar', 'since', 'skin', 'something', 'sort', 'sound', 'spice', 'star', 'starters', 'things', 'thought', 'today', 'ultimately
         'us', 'use', 'vaseline', 'way', 'would']

In [26]: import numpy as np
         dist = np.sum(train_data_features, axis=0)

         for tag, count in zip(vocab, dist):
             print(count, tag)

         2 absolutely
         1 affect
         1 album
         1 based
         1 bedroom
         4 bought
         2 boyfriend
         1 burning
         1 buy
         1 buying
         2 captivating
         1 caused
         1 check
         1 clean
         1 collected
         1 consistency
         2 could
         1 couples
         1 crazy
         1 current
         1 description
         1 difficult
         4 disappointed
         1 earlier
         1 enhanced
         1 especially
         1 even
         1 everyday
         1 excited
```

Fig. 9. Bag of Words

Source Code : https://github.com/Jinansi/PythonProjects/blob/master/ADS_Project/BagOfWords.ipynb

5. Deep Learning Algorithm:

a.) Recurrent Neural Network:

It designed to recognize patterns in sequences of data, such as text, genomes, handwriting, the spoken word, or numerical times series data emanating from sensors, stock markets and government agencies.

We have implemented RNN to predict rating. We have added LSTM layer to get better accuracy with relu activation function.

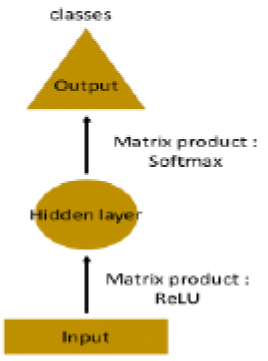
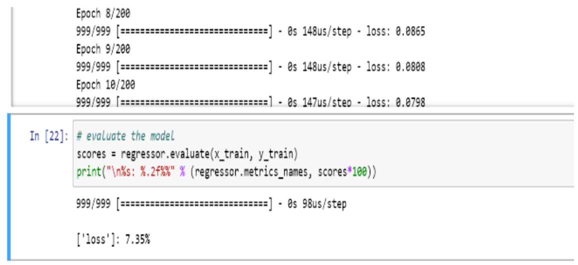
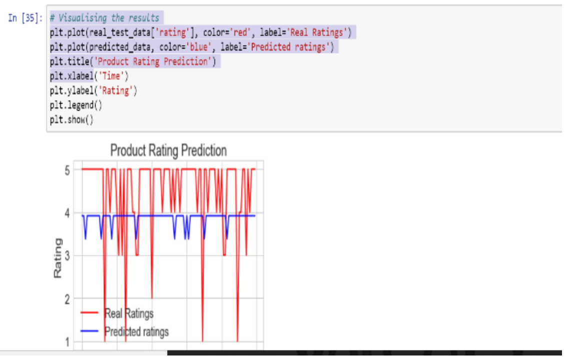


Fig. 10. Recurrent Neural Network Working



```
Epoch 8/200
999/999 [==============================] - 0s 148us/step - loss: 0.0865
Epoch 9/200
999/999 [==============================] - 0s 148us/step - loss: 0.0808
Epoch 10/200
999/999 [==============================] - 0s 147us/step - loss: 0.0798

In [22]: # evaluate the model
         scores = regressor.evaluate(x_train, y_train)
         print("\n%s: %.2f%%" % (regressor.metrics_names, scores*100))

         999/999 [==============================] - 0s 98us/step

         ['loss']: 7.35%
```

Fig. 11. Total loss while evaluating model



```
In [35]: # Visualising the results
         plt.plot(real_test_data['rating'], color='red', label='Real Ratings')
         plt.plot(predicted_data, color='blue', label='Predicted ratings')
         plt.title('Product Rating Prediction')
         plt.xlabel('Time')
         plt.ylabel('Rating')
         plt.legend()
         plt.show()
```

Fig. 12 Recurrent Neural Network Result

The above product shows the predicted and actual values of rating.

Source code : https://github.com/Jinansi/PythonProjects/blob/master/ADS_Project/RNN_Final.ipynb

IV. CONCLUSIONS

By building this recommendation system, we wanted to find products which are similar and can be recommended to a set of users who have similar buying patterns. Moreover, by using algorithms like content based filtering, we try to find similar reviews given by multiple users. After using various algorithms, we came to the conclusion that Random Forest, RNN (Recurrent Neural Networks), Content-Based Algorithm, Bag of Words are very much suitable for our dataset and for performing the required analysis whereas ALS (Alternating Least Square) serves as a fair one. Naive Bayes and k-means clustering are not suitable for the same since accuracy is way too way for it to qualify in our project.

V. REFERENCES

- https://github.com/nikbearbrown/NEU_COE/blob/master/INFO_7390
- https://chrisalbon.com/machine_learning/trees_and_forests/random_forest_classifier_example/
- https://towardsdatascience.com/recommender-engine-under-the-hood-7869d5eab072
- https://machinelearningmastery.com/deep-learning-bag-of-words-model-sentiment-analysis/
- https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/
- https://www.kdnuggets.com/2016/01/seven-steps-deep-learning.html/2
- https://www.ibm.com/developerworks/library/os-recommender1/